ADMI: A Multi-Agent Architecture to Autonomously Generate Data Mining Services

Syed Zahid Hassan Zaidi, S. S. R. Abidi, S. Manikam & Cheah Yu-N

Abstract—**This paper presents a case for an intelligent agent based framework for knowledge discovery in a distributed environment comprising multiple heterogeneous data repositories. Data-mediated knowledge discovery, especially from multiple heterogeneous data resources, is a tedious process and imposes significant operational constraints on end-users. We demonstrate that autonomous, reactive and proactive intelligent agents provide an opportunity to generate end-user oriented, packaged, value-added decision-support/strategic planning services for professionals, managers and policy makers of an organization, without the need for a priori technical knowledge. Since effective progress of an organization is grounded in good communication, experience sharing, continuous learning and proactive actions, we present an Agent-based Data Mining Info-structure (ADMI) that deploys a suite of Data Mining (DM) algorithms coupled with intelligent agents to facilitate data access, DM query specification, DM algorithm selection and DM result visualisation—i.e. automated generation of data-mediated decision-support/strategic-planning services.**

Index Terms— **Agents, Multiagent systems, Multiagent methodology, Data mining and Healthcare**

I. Introduction

Data Mining is a prevalent activity in a variety of domain whereby the operational data collected is used to provide insights into the workings of the enterprise and to use the gleaned information/knowledge to both strategise and improve performance and outcomes.

Dynamically-generated data mediated services are designed to provide strategic insights, as per a user request, from static operational data. The vantage point of these services are that they provide insights to assists healthcare analyst and policies makers to make strategic decisions or predict future consequences by taking into account the actual outcomes of current operative values. In a healthcare context, typical services may include: Analysis, planning, trending, examine, forecasting, predicting bench marking and best practices reporting, outcomes measurement, what–if scenario analysis, comparing organization practice with organization rules, market research , effectiveness on outcomes of treatment, data analysis for organization financing, health surveillance and resource allocations.

Manually, the generation of services is a non-trivial exercise as it involves a sequence of tasks such as, the analysis of the user's request, selection of the data pertinent to the request [1], selecting and deploying an apt data mining method and finally presenting the results to the user. A preferable way of system working is to construct agent wrappers around KDD (knowledge discovery in databases) systems [2]. These agent wrappers interface to the information sources and information consumer, providing a uniform way to accessing data as well as offering additional functionalities, such as monitoring the changes and provide the services on demand. The critical question then is how to structure and organize these multiple agents to achieve user-specific dynamic data-driven services.

In our work, we investigate the dynamic and autonomous generation of data-driven services in response to a user request for a service. We present a data-mediated services system that builds on multiple intelligent agents to dynamically generate user-specific services. In this paper we present a an Agent-based Data Mining Info-structure (ADMI) that deploys a suite of Data Mining (DM) algorithms coupled with intelligent agents to facilitate data access, DM query specification, DM algorithm selection and DM result visualisation—i.e. automated generation of data-mediated decision-support/strategic-planning services. ADMI is designed to service four functional components—(i) end-user interface; (ii) remote data access network; (iii) data mining engine; and (iv) strategic services—and hence comprises of multiple autonomous intelligent agents such as Interface agent, Data Mining Agent, Service Generation agent, Data Collection Agents and Agent Manager.

The aim of this paper is to introduce the intelligent agent based paradigm for distributed Data Mining (DM) from heterogeneous data repositories. The features of ADMI is its ability to realize a dynamic agent organization in which each individual agent acts to meet the user's request whilst hiding the data mining complexities from the user. This is accomplished by designing each ADMI agent with specific plan file and knowledge bases (Task schemata) which

Syed Zahid Hassan Zaidi is with the Health Informatics Research Group, School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia (telephone: 0060-125351262, email: zahid@cs.usm.my)

S. S. R. Abidi, is with the Faculty of Computer Science, Dalhousie University, Halifax B3H 1W5, Canada (telephone: 1-902-4942129 email: ssrabidi@cs.dal.ca)

S. Manikam is with the Health Informatics Research Group, School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia (telephone: 0060-125351262, email: selva@cs.usm.my)

Cheah Yu-N, Health Informatics Research Group, is with the Health Informatics Research Group, School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia (telephone: 0060-125351262, email: yncheah@cs.usm.my)

0-7803-8278-1/04/$20.00 ©2004 IEEE

allow agents to perform their tasks and knowledge discovery activities without being monitored and controlled by the user. Functionally, the different agents collaborate, corporate and interoperate with each other through a central middle agent. The middle agent provides look up services (yellow pages) to identify agents with their specific capability description and abetting them to communicate each other. ADMI has been developed to provide strategic services in a healthcare context and we use the same context in our discussion.

II. Multiagent Systems— A Brief Overview

There are diversify number of methodologies have been proposed and various design tools available for the design of multiagent systems. The development of that design tools range from commercially available products to research prototypes. We briefly focus first on some available advanced and widely used tools kits for multiagent systems designing and then compare our propose ADMI infrastructure with some those existing multiagent systems.

AgentBuilder: AgentBuilder is a commercial product produced by Reticular Systems, Inc. It provides an integrated toolkit for the construction of intelligent agents and provides various graphical tools including project management, ontology management, agent management, protocol management, and run time engines. AgentBuilder is Java-based and its communication language is KQML. The AgentBuilder agent model consists of agents that have beliefs, capabilities, commitments, intentions, and behavioral rules [3].

ZEUS: developed at British Telecom Laboratories by Collis et al., is an advanced toolkit for engineering distributed multi-agent systems. ZEUS specifies a design approach utilizing a three-layer model of an agent, the definition layer, the organization layer, and the coordination layer. It provides a set of graphical interfaces to allow a designer to create the multi-agent system. An agent is comprised of libraries that include communication (using KQML), social interaction, multi-agent coordination, and planning and scheduling. This tool generates source code for the agent-based system in Java [4].

DECAF: Distributed Environment-Centered Agent Framework is a software toolkit for developing intelligent agents. It provides user interfaces for the specification of the agent behavior, including logical and temporal constructs. It allows reuse of generic agent behaviors. Its Plan-Editor removes the designer of a multi-agent system from the low level API and allows for the rapid development of agents, which are generated in Java. DECAF also provides some design time error checking, and supports the use of agent communication languages [5]. *JADE:* JADE (Java Agent Development Framework) is a software framework for developing FIPA-compliant multi-agent systems. JADE provides a set of interfaces for the design of agents, implemented in Java. JADE uses the FIPA-ACL, utilizing a combination of socket, RMI and CORBA [6].

The Gaia Methodology: It presents a methodology for agent-oriented analysis and design. Using Gaia, software designers can systematically develop an implementation-ready design based on system requirements. The first step in the Gaia analysis process is to find the roles in the system, and the second is to model interactions between the roles found. In the Gaia design process, the first step is to map roles into agent types, and then to create the right number of agent instances of each type. The second step is to determine the services model needed to fulfill a role in one or several agents, and the final step to create the acquaintance model for the representation of communication between the agents [7].

NZDIS: The New Zealand Distributed Information Systems (NZDIS) project at the University of Otago is a tool to develop new agent-based technology and techniques that can be used to provide integrated access to disparate information sources distributed across the Internet. The goal is to take advantage of existing, standard object-oriented approaches, wherever possible, and combine them with newly developed agent-oriented techniques [8].

RETSINA: It consider MAS infrastructure to be the domain independent and reusable substratum on which MAS systems, services, components, live, communicate, interact and interoperate, while the single agent infrastructures, the generic parts of an agent, enable it to be part of the a multiagent society. Each RETSINA agent has four reusable modules for communicating, planning, scheduling and monitoring the execution of tasks and requests from other agents. A RETSINA agent is distinguished according to the kind of task it performs (i.e. interface, task, and information agents) and successfully be applied in many domains ranging from financial portfolio management to logistic planning [9].

B. ADMI Design and Methodologies— Comparing with Existing MAS Frameworks

The design and methodology of ADMI framework is based on above discussed MAS frameworks [5, 5, 7] with some modification/amendments/extension with RETSINA methodology [9]. ADMI deploys Gaia's schemata [7], for the standard representation of tasks and associations among the tasks and Retsina's schemata [9], as a standard way to face such challenges ahead; (1) how to decompose problems and allocate tasks to individual agents; (2) how to coordinate agent control and communications; (3) how to make multiple agents act in a coherent manner; (4) how to make individual agents reason about other agents and the state of coordination; (5) how to reconcile conflicting goals between coordinating agents; and (6) how to engineer practical multi-agent systems.

The above-mentioned design tools use various approaches to facilitate the design of multi-agent systems (software components). Among these, there are some aspects that are notably similar to ADMI and other areas such as agent conversation, actions and knowledge representations, planning of agents and maintain current state of agents, where ADMI improves on the current collection of work. Since most of the advanced and tested multiagent framework such as Decaf [5], NZDIS [8], and ZEUS [4] etc., deploying Retsina's [9] generic agent's software components: Communication, Objective Queue, Planner, Task Queue, Scheduler, Execution Monitor and

Local Database, to their applications. Thus, ADMI framework also adopts all the basic software components (modules) designed in Restina's generic agent architecture with some necessary modification.

Retsina provides the mechanism for agents to interact each other but there is no such standard mechanism to maintain the current state of agent conversation dialogues which is important in some situations such as, in case of any talking/communicating agent die or failure or if any agent also want to talk to the same agent who is busy in talking with other agent. Checking the conversation models for consistency and coherency is not a feature that many design tools have, although DECAF does have some design time error checking and JADE [6], provides some simulation and debugging capabilities. To address this issue, ADMI introduces Separate conversation module which maintain the state of one of the agent's current conversation dialogue in accordance with a conversation policy appropriate to agent's role in that conversation [8]. On the execution of any action or plan RETSINA does not provide any mechanism to keep track/store/record/maintain the current state of same agent within its models. In some cases, it's necessary for an agent to remember its capabilities and its current state; what action is being executed and what has previously been performed. On the other hand, ADMI has separate memory component similar to NZDIS memory module where the current state of the agent can be stored in declarative form.

ADMI like other multiagent systems or design tools has turned to Java to provide platform independence. This choice affects the deployment of the design tools themselves as well as the systems that these tools design. ADMI is written in Java and produces Java based multi-agent systems. Notably, all ADMI agents' modules are designed in separate autonomous thread (multithreaded), and thus all modules execute concurrently and continuously except for agent initialization [5, 9].

III. Our Proposed Framework

In this section, we present the architectural and functional overview of our proposed multi Agent-Based Data Mining Info-Structure (ADMI), as depicted in figure 1. The system takes advantage of the power of multi-agent architecture which is the amalgamation of various types of intelligent agents, each of them handling services among the agents, databases or end-users. The agent-federation is designed to service four functional components—(i) end-user interface; (ii) remote data access network; (iii) data mining engine; and (iv) diagnostic-support and strategic services—and hence comprises of five types autonomous intelligent agents which are described below.

Interface Agents: The Interface Agent (IA) manages a web-based graphical interface for the specification of a data-mediated service by the user. More specifically, the IA takes as input a set of service goals and autonomously translates them to three follow-up task specifications—(i) the DM tasks that need to be performed, the data that need to be retrieved and (iii) the results (i.e. the service) presentation style.

Data Collection Agents: The core functionality of the Data Collection Agent (DCA) is to facilitate the on-demand retrieval of 'relevant' data from the multiple healthcare data repositories. In an autonomous manner, DCA performs the following tasks: (a) establishing protocols for remote data access; (b) data selection and retrieval; (c) and data synthesis.

Data Mining Agents: The Data Mining Agent (DMA), as its name suggests, is responsible for coordinating the entire DM activities. The DM tasks are classified into four classes: classification, prediction, association, clustering, where each class may represent multiple DM algorithms. Each DM algorithm is implemented as a DM module. Given a task specification from the IA, the DMA autonomously selects the most appropriate DM module and coordinates all its processing requirements—i.e. data, constraints, protocols and so on.

Services Generation Agents: The Services Generation Agent (SGA) processes the DM results produced by the DMA to generate decision-support/strategic services as per the user's request.

Agents' Manager: It is an agent with high computational power that acts as both the point of control for the distributed data mining activities and the provision of dedicated resources for mining. It forms the core of ADMI infrastructure, the way it structured, encapsulates the frame work for reliable resource optimization. The main functions of this agent are to control coordinating/calloborating/negotiating facilities among the agents, including mining process management and discovery constraints, and maintaining the communication protocols to the system working.

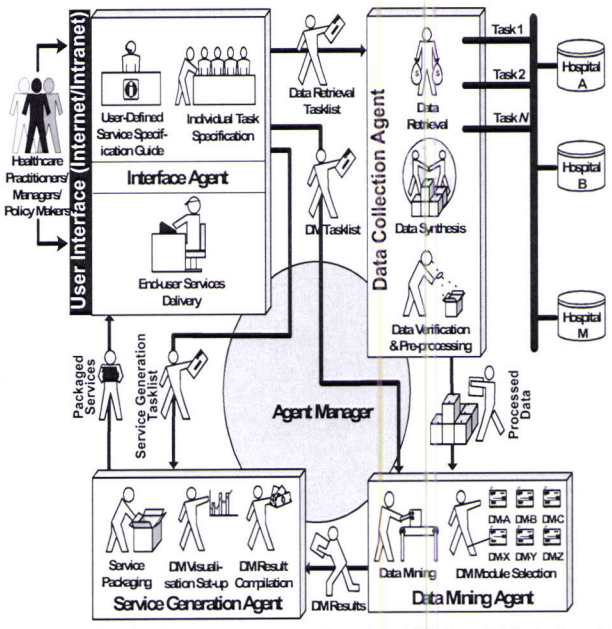

Fig. 1. A sketch of an agent community—i.e. ADMI—which deals with the DM task given in scenario. Illustrated is the intrinsic behavior and inter-agent communication of the inherent intelligent agents.

The technical workflow of ADMI illustrates the interaction of various agents in accomplishing the task of converting the user query into graphically visualized results. The algorithm mainly entails the following:-

- Getting query from the user and generating a list of possible fields from the query.
- Based on the list of fields, various table names are

retrieved from the databases that are distributed on various computers.

- The retrieved data is the formatted and fed in the appropriate data mining engine based on user requirements.
- Finally, the result generated by the data mining engine is then shown to user using intuitively generated graphical charts. Figure 3, illustrates the overall technical workflow of ADMI.

Figure 2, shows the confluence of technologies used to implement the abovementioned workflow. Figure 3, illustrates the overall technical workflow of ADMI.

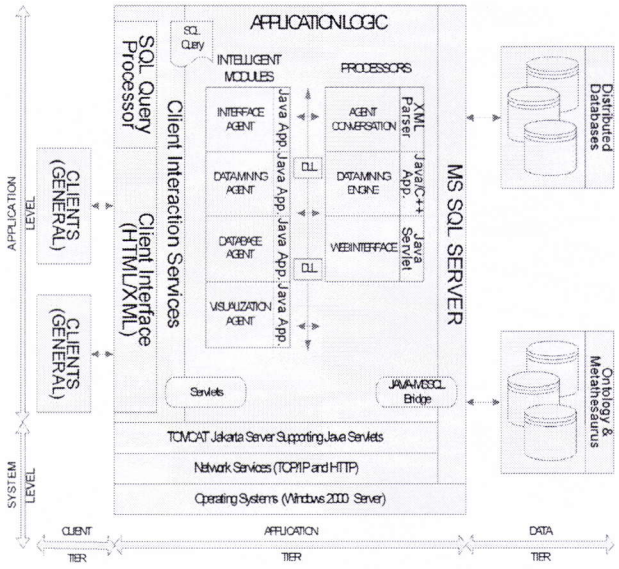

Fig. 2. Technical architecture of ADMI illustrating the various development platforms.

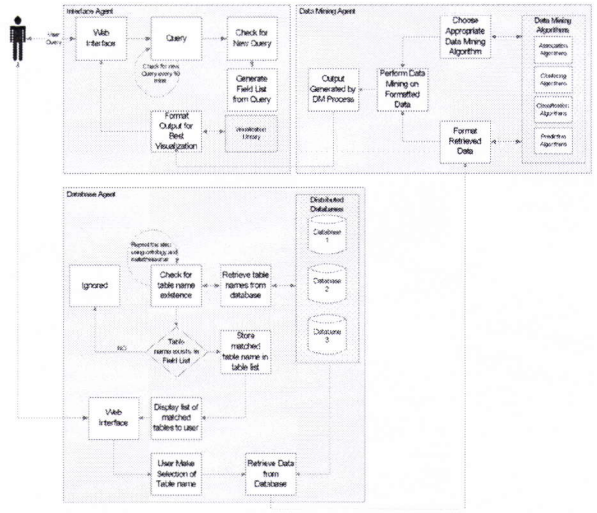

Fig. 3. Technical workflow of ADMI illustrating interaction of various technologies.

IV. Agent Task Representation Schema

The design and development of ADMI framework is incorporated by GIA multiagent BDI based methodology [7], where the entire multiagent system has been captured as an organisation and the individual agents as the particular role/member of the organization. In our case, we view the ADMI as an organization—a collection of roles, that stand in certain relationships to one another, and that take part in systematic, institutionalized patterns of interaction with other roles. Each agent is assigned with a particular role, limited number of tasks and actions in the framework.

Each role is defined with four standard attributes: *responsibility*, *permissions*, *activities*, and *protocols* to perform appropriate actions and then its task/action is represented and achieved by *Task Schema* and *Task Reduction Schema* [2].

Responsibility roles' functionality of an agent is measured by the responsibility assigned to it, which is divided into three categories: liveness, safety and security. The liveness property ensures the "task will be done" by performing certain actions. For illustration purposes, we discuss the monitoring (liveness) responsibility of data collection agent/role. The common liveness property of that agent is to "inform the related agent whenever new information is being updated in data resources". This property can be specified by using the liveness expression as:

$$RoleName = Expression$$

Where the liveness expression constitutes both activities and/or protocols. The information agent role is therefore given as follows, and expresses that the DataMonitor consists of execution protocol Monitor, followed by the protocol informAgent, followed by the activity CheckStock and a protocol AwaitUpdate.

$$DataMonitor = (Monitor.InformAgent, CheckStock. AwaitUpdate)$$

By convention, the role model can technically be defined; a role model is comprised of a set of role schemata, one for each role in the system.

Role Schema	**Name of Role**
Description	Short English description of the role
Protocols and Activities	In which the role plays a part
Permissions	Rights associated with the role
Responsibilities	
Liveness	liveness responsibilities
Safety	safety responsibilities.

Fig. 4. Template for Role schemata

Permissions are the "rights" associated with the roles to realize their responsibility. Protocols define the mechanism for the roles to interact with each other. Figure 4, & 5, shows the standard way to represent schemata for role DataMonitor which is the one of the task of data collection agent.

Role Schema: DataMonitor		
Description: This role involves ensuring that database is kept recorded and informs the agents in case of any updating.		
Protocols and Activities: *Monitor, informAgent, CheckDatabase* and *AwaitUpdate.*		
Permissions:	reads	supplied *DataProvider* *DataStatus*
	changes	*StockData*
Responsibilities Liveness: *DataMonitor= (Monitor.InformAgent. CheckDatabase. AwaitUpdate)* Safety: *StockData > 0*		

Fig. 5. Standard Schema for role DataMonitor

V. Agent's Task Representation

In our agent framework, each task is formally represented by a tuple $<N,Par,Dpar,Pro,Out,C,E>$, where N is the name of the task. Par and $Dpar$ are the static and dynamic parameters respectively that represent beliefs of the agent. The static parameter is known to the agent at a planning time and dynamic parameter is known only at an execution time of the plan. Pro, dynamic provision parameters, is a set of provisions are used to describe the set of flow of information in the plan. Out — is a set of outcomes of the task. Finally C is a set of constraints which defines the condition under which task is to be executed and E, is a set of estimators used by the planner to predict the outcomes of the task. Figure 6, gives an exemplar representation for the task 'patient treatment'.

Patient Treatment

- Patient problem
- Branch of hospital
- Patient disease
- Enable
- Medicines, operation etc
- Symptoms

Treatment suggested

Fig. 6. In this example, N = [patient treatment], Par = [Patient problem, branch of hospital], Dpar =[patient disease], Out = [treatment], C = [medicines, operation etc.], E = [symptoms]

For purposes of explication we present the Data Collection Agent Schema for the DataMonitor Task as depicted in figure 7.

VI. A Working Example

The user interaction with the system is a seaming less and allows the user to specify only the kind of data and analysis to be performed. The rest of the tasks are handled by the various agents working together to accomplish the task in conjunction with the user. The interaction flow is illustrated in sequence as following using screenshots from the screen, as shown in figure 8-13.

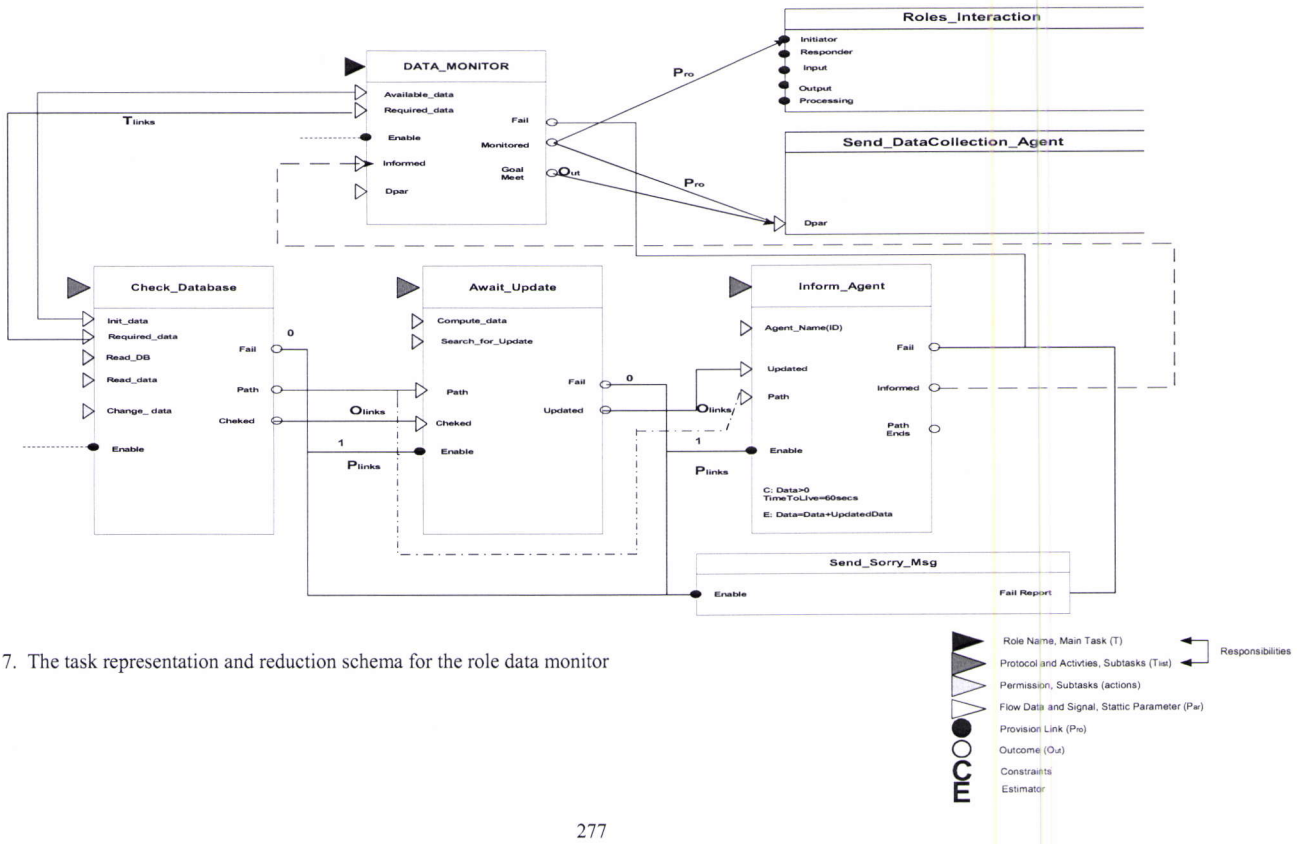

Fig. 7. The task representation and reduction schema for the role data monitor

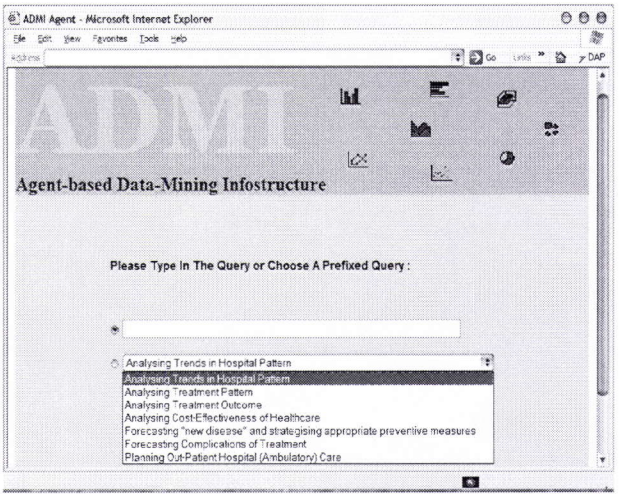

Fig. 8. Step 1. User chooses the analysis to be performed or custom analysis can also can be specified

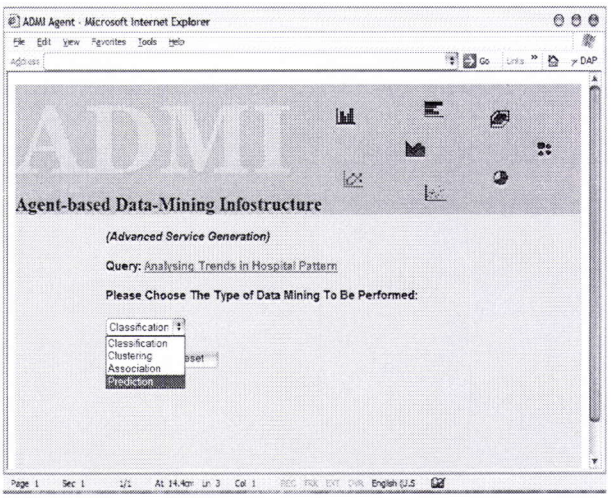

Fig. 9. Step 2. Based on the chosen analysis, the user specifies the type of Data Mining to be performed.

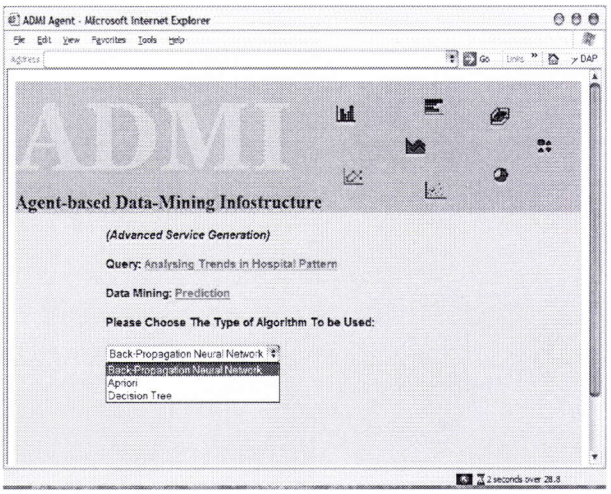

Fig. 10. Step 3. User would then choose the specific algorithm data mining algorithm to be performed.

VII. Conclusion

Data can be analyzed from different perspectives, each perspective imparting a different kind of knowledge— for instance hospital admissions data can be used to both

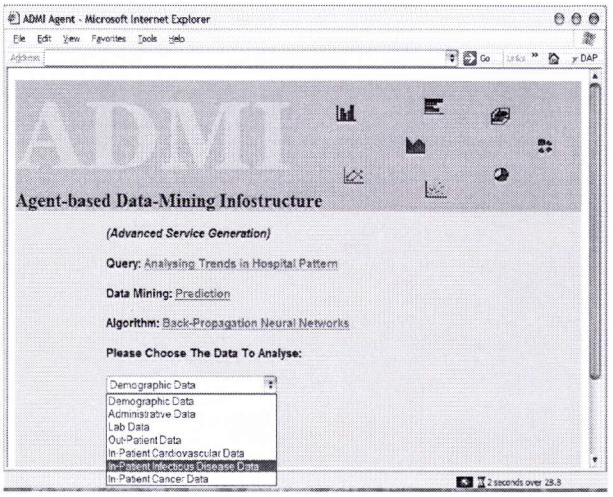

Fig. 11. Step 4. Data is chosen for the appropriate data mining algorithm.

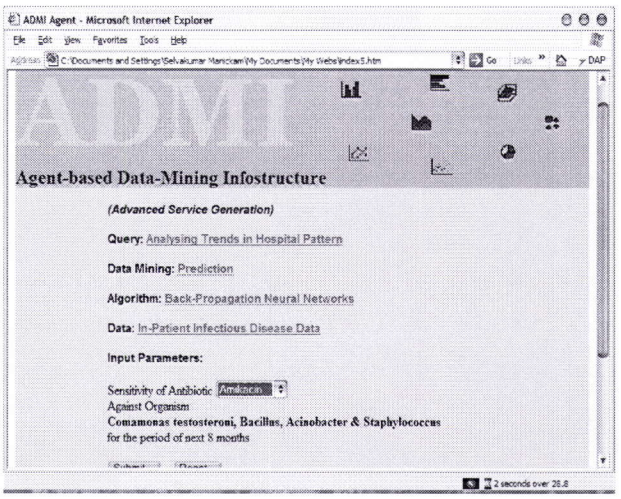

Fig. 12. Step 5. Summary of the whole process is displayed for verification and data mining is performed.

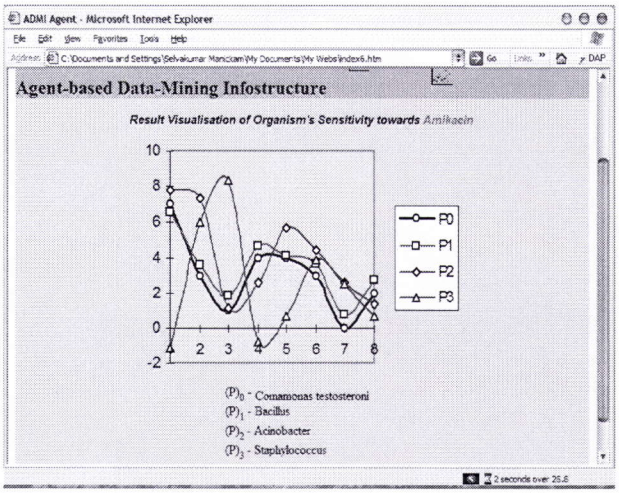

Fig. 13. Step 6. Result generated by the data mining algorithm is displayed using suitable visualization method.

analyze hospital admissions and also to forecast future hospital admission. In our opinion, multiple usage of data can be supported by the collaborative use of multiple, task-specific agent communities. The fact that all agents communicate through a common language, allows for an

open-ended architecture whereby new DM services and DM algorithms can incrementally be added. This implies the need to take into account, and leverage as much as possible, the results of multiple intercommunicating agents, each working on some aspect of the data. In this paper, we have demonstrated that the multiple heterogeneous data repositories in an enterprise need to establish a distributed data community, such that any DM effort draws upon the 'holistic' data available within the entire enterprise. When adopting this view, a set of data access and mining issues can be addressed using the well-known software agent technology. Thus far, in our work, we have managed to develop prototype ADMI architecture to leverage intelligent agents for generating (healthcare) data-mediated decision-support/strategic-planning services targeted from healthcare professionals, managers to policy makers.

References

[1] S.S.R. Abidi, Y-N., "A convergence of Knowledge management and data mining: towards 'knopwledge driven' strategic services," 3rd Int. Conferennce on the practical applications of knowledge management (PAKeM'2000), 2000, Manchester.

[2] Khaled A. Arisha, "Impact: A Platform for Collaborating Agents", IEEE Intelligent Systems, vol. 14, no. 2, pp. 64-72, March/April 1999.

[3] AgentBuilder Web site. "Agent construction tools". Available at http://www.agentbuilder.com/AgentTools/index.html.

[4] Lee L. C., Ndumu D. T., Nwana H. S., "ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems." In Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Systems, pp.377-392, Londres, 1998.

[5] Decker, K.; Pannu. A.; Sycara, K; and Williamson, M. 1997. "Designing Behaviors for Information Agents." In Proceedings of the First International Conference on Autonomous Agents (Agent-97), 404-412. New York:Association of Computing Machinery.

[6] Bellifemine, F., Poggi, A., and Rimmassa, G., "JADE - A FIPA-compliant agent framework." CSELT internal technical report. Also published in part in the Proceedings of PAAM '99, April 1999, pp. 97-108.

[7] Wooldridge M. J., Jennings N. R. and Kinny D. "The Gaia Methodology for Agent-oriented Analysis and Design." Autonomous Agents and Multi-Agent Systems, 3(3):285–312, September 2000.

[8] Martin Purvis, Stephen Cranefield, Geoff Bush, Daniel Carter, Bryce McKinlay, Mariusz Nowostawski, and Roy Ward. "The NZDIS Project: an Agent-based Distributed Information Systems Architecture." In R.H. Sprague Jr., editor, CDROM Proceedings of the Hawaii International Conference on System Sciences (HICSS-33). IEEE Computer Society Press, 2000.

[9] Sycara K., Paolucci M., Velsen M. V., Giampapa J., "The RETSINA MAS Infrastructure."

[10] J. Han, Y. Fu, et al, 1996. "DB Miner: A System For Mining Knowledge in Large Relational Databases." In Int'l Conf. on Data Mining and Knowledge Discovery (KDD'1996), Portland, Oregon, August 1996, pages 250-255."

[11] Busetta P., Ronnquist R., Hodgson A., Lucas A., "Jack Intelligent Agents — Components for Intelligent Agents in Java," AgentLink News Letter, Janvier 1999.

[12] Chauhan D., "JAFMAS: A Java-Based Agent Framework for Multi-Agent Systems Development and Implementation," Masters Thesis, ECECS Department, University of Cincinnati, July 1997.

[13] DeLoach S., A., Wood M., "Developing Multiagent Systems with agentTool," (ATAL'2000), Berlin, 2001.

[14] Martin D., Cheyer A., Moran D., "The Open Agent Architecture: A Framework for Building Distributed Software Systems," Applied Artificial Intelligence, vol. 13, no 1-2, pp 91-128, June 1999.

[15] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an Agent Communication Language," Software Agents, Menlo Park, Calif.: AAAI Press, pp. 291-316, March 1997.

[16] H. Kargupta, I. Hamzaodlu, and B. Stafford, "Sclable, Distributed Data Mining Using an Agent Based Architecture", D.Heckerman, et al (Eds) Proc. of 3rd Int.Conf. on knowledge discovery and Data Mining, AAAI press, pp.211-214, June 1997.

[17] B. Hayes-Roth and J. E. Larsson, "A domain-specific software architecture for a class of intelligent patient monitoring systems", J. Experimental Theoretical Artificial Intelligence, vol. 8, pp. 149–171, July 1996.

[18] S. S. R. Abidi, "Applying Knowledge Discovery in Healthcare: An Info-Structure for Delivering 'Knowledge-Driven' Strategic Services", P. Kokok et al. (Eds) Medical Informatics Europe'99, Ljubljana, Amsterdam:IOS Press, pp. 453-456, Feburary 1999.

IS'04
IEEE International Conference 'Intelligent Systems'
Bulgarian Council of Ministers Holiday Complex, St.St. Constantine and Helena Resort, Varna, Bulgaria, June 22-24, 2004

IS'04 PROGRAM AND CALL FOR PAPERS

The Conference will provide a forum of research, development, and applications of intelligent systems techniques across a variety of disciplines. All aspects of intelligent systems are of interest: tools, theory, algorithms, applications, etc.

Topics of interest

Methodologies:

Artificial intelligence
Data mining
Decision support systems
Fuzzy logic
Intelligent control
Intelligent systems and semiotics
Machine learning
Neural networks
Chaos theory
Data fusion
Evolutionary computation
Human-machine interaction
Intelligent measurement
Knowledge engineering
Neuro-fuzzy systems
Soft computing agents

Application areas:

Automotive
Educational aspects of intelligent control
Robotics
Telecommunications
Data processing
Process control
Transportation
etc.

Sponsors and supporters:

IEEE Instrumentation and Measurement Society

IEEE Section Bulgaria

IEEE IM/CS/SMC Joint Chapter of Bulgaria

Bulgarian Federation of Technical Unions

Institute of Information Technologies of the Bulgarian Academy of Sciences

Union on Automation and Informatics of Bulgaria.

Honorary Chairman of IEEE IS'04: L.A. Zadeh, United States of America
Conference Co-Chairs: R. Yager, United States of America
V. Sgurev, Bulgaria
Chair of Organizing Committee: M. Hadjiski, Bulgaria
General Coordinator: V. Jotsov, Bulgaria

ORGANIZING COMMITTEE

K. Alexiev
K. Atanassov
E. Bojanov
M. Grigorova
V. Jotsov
P. Koprinkova
S. Koynov
O. Manolov
K. Stoilova
K. Tenekedjiev
F. Tomova
S. Vollosovitch

INTERNATIONAL PROGRAM COMMITTEE

P. Albertos, Spain
P. Antsaklis, United States of America
J. Austin, United Kingdom
R. Babuska, The Netherlands
J-P Bourrieres, France
I. Bratko, Slovenia
Z. Bubnicki, Poland
F. Capkovic, Slovakia
V. Cherkassky, United States of America
A. Cipriano, Chile
G. Dimirovski, Turkey
A. Dinibutun, Turkey
D. Driankov, Sweden
N. Durakbasa, Austria
D. Filev, United States of America
G. Gatev, Bulgaria
V. Gladun, Ukraine
A. Glattfelder, Switzerland
P. Groumpos, Greece
M. Hadjiski, Bulgaria
K. Hirota, Japan
G. Horvath, Hungary
G. Irwin, United Kingdom
L. Jain, Australia
P. Jorrand, France
J. Kacprzyk, Poland
N. Kasabov, New Zealand
O. Kaynak, Turkey
V. Kecman, New Zealand
G. Klir, United States of America
A. Kordon, United States of America
D. Lakov, Bulgaria
R. Langari, United States of America
K. Leiviska, Finland
D. Linkens, United Kingdom
D. Matko, Slovenia
K. Narendra, United States of America
P.H. Osanna, Austria

R. Palm, Germany
T. Parisini, Italy
T. Samad, United States of America
A. Shannon, Australia
V. Stefanuk, Russia
S. Strmcnik, Slovenia
H. Tamura, Japan
M. Thuillard, Switzerland
S. Tzafestas, Greece
A. Vasilakos, Greece
M. Vukobratovic, Serbia and Montenegro
H. Wang, United Kingdom
K. Warwick, United Kingdom

Prospective authors are solicited to send via E-mail or via the post a draft paper (maximum 6 pages) for review. All submissions must be written in English, PDF format. The rules for the authors are described in the conference WEB site. The submissions must be sent via E-mail to pkoprinkova@icsr.bas.bg

The submissions could be sent also via airmail to:

IS'2004, P.O.Box 161, Sofia 1113 Bulgaria
(including 1 copy and the electronic version). **In this case** it will be additionally checked on the accordance with the demands for the IEEE conference camera ready papers.

The expression of interest should be sent to pkoprinkova@icsr.bas.bg

Poster presentations are encouraged for people who wish to receive peer feedback. All accepted papers will be published. The organization of invited sessions is also encouraged.

The submissions have to include:
Title of the conference topic;
Type of paper (invited, contributed, poster or tutorial);
Title of proposed paper;
Authors names, affiliations, address;
Maximum 5 keywords describing the topic.

The authors E-mail address should be included.

IMPORTANT DATES

are given at the conference site

Secretary of IEEE IS'04: P. Koprinkova. Phone: +359 2 870 03 37.
Email: pkoprinkova@icsr.bas.bg